

MC EPROM SET

C O N T E N T S

CONTENTS i

INTRODUCTION ii

INSTRUCTIONS 1

FUNCTION or SUBROUTINE	DESCRIPTION	LISTING
BURNL.	1	6
BURNU.	2	7
GETEP.	2	19
ROM>REG.	3	17
REG>ROM.	3	12
READROM.	3	11
WRTROM	4	14
RCOPY	5	10
PCAT	5	24
KASN	5	22
LASTREG.	-	18
INIT MC000506A . . .	-	25
ILDATA	-	27
DISPNAME	-	28
ADR2DIS.	-	28
BCDBIN	-	29
BCDBIN X	-	29
BCDBIN XF.	-	29
BINBCD	-	30
FINDID	-	31
ERRMSSG.	-	32
PILTST	-	32
ERRHLT	-	33

MC EPROM SET

for use with

Mountain Computer MC00506A EPROM Programmer

1). This software package allows HP-41 formatted EPROMs to be programmed using the Mountain Computer MC00506A HPIL EPROM Programmer. The ROM to be programmed must reside in the ROM address space of the HP-41. This can be a ROM module, another EPROM set (using an EPROM box), or a 4k block of RAM (in an MLDL type of device).

2). The EPROM set and its source code was placed in the public domain July 9, 1983 by Paul Lind, and may be copied, distributed, or modified in any way you see fit.

3). This software package is not an official product of Mountain Computer, Inc., and must be considered NOMAS (Not Manufacturer Supported).

4]. Copies of this EPROM set are available through PPC. Cost for manual and EPROM set is $20.00.* (1-2716, 1-2732). Order From:

PPC - POB 9599
Fountain Valley, CA 92728-9599
USA

*USA Price. Add shipping and 6% sales tax if shipped to a California address. Shipping weight is 6 oz.

INSTRUCTIONS FOR
MOUNTAIN COMPUTER EPROM SET

*** EPROM FUNCTIONS ***

BACKGROUND ON HP-41 EPROMS

The HP-41 ROM space uses a 10 bit word size. To store these ROM words in standard EPROMs (byte-wide), it was decided to use separate EPROMs for the lower eight bits and the upper two bits. Each byte of the L8 EPROM contains the lower eight bits for one of the 10 bit ROM words. However, to conserve space, each byte of the U2 EPROM contains the upper two bits for FOUR consecutive 10 bit ROM words. Therefore, burning a 4k HP-41 ROM block requires 4k of L8 EPROM space, but only 1k of U2 EPROM space.

Since a 2716 (the minimum size U2 EPROM) is 2k long, there is room for the U2 information from two separate HP-41 4k blocks. If a 2732 is used as the U2 EPROM, there is room for the U2 information from four separate HP-41 ROM blocks. So that the user does not have to remember EPROM addresses, the programs that burn or read EPROMs ask for a section number. For example, a section number of 0 specifies that the lowest 1k area in a U2 EPROM. L8 EPROMs also use section numbers, since a 2764 may store the L8 information from two separate 4k blocks.

BURNL

FUNCTION: Program an L8 EPROM from an existing 4k HP-41 ROM block, using the MC00506 EPROM Programmer. The first MC00506 found on the loop starting with the primary device is used. The program may be terminated by pressing the "ON" key.

INPUT: Y REG - The section number to burn (0 - 1).
X REG - The ROM block to burn from (0 - 15).

OUTPUT: NONE

ERRORS:

DATA ERROR	Block number) 15 or section number) 1.
ALPHA DATA	Block number or section number is type ALPHA rather than NUMERIC.
NO HPIL	HPIL module is not plugged in.
NO MC00506A	EPROM Programmer not on loop or loop in MANIO and MC00506A not the primary device.
ERASE ERR	Specified section of EPROM not erased.
BURN ERR	Byte not programmed successfully.
LOW BAT ERR	The HP-41 battery voltage dropped too low to continue reliable operation.
TRANSMIT ERR	Loop not operating correctly.

BURNU

FUNCTION: Program a U2 EPROM from an existing 4k HP-41 ROM block, using the MC00506A EPROM Programmer. The first MC00506A found on the loop starting with the primary device is used. The program may be terminated by pressing the "ON" key.

INPUT: Y REG - The section number to burn (0 - 3).
X REG - The ROM block to burn from (0 - 15).

OUTPUT: NONE

ERRORS:		
	DATA ERROR	Block number > 15 or section number > 3.
	ALPHA DATA	Block number or section number is type ALPHA rather than NUMERIC.
	NO HPIL	HPIL module is not plugged in.
	NO MC00506A	EPROM Programmer not on loop or loop in MANIO and MC00506A not the primary device.
	ERASE ERR	Specified section of EPROM not erased.
	BURN ERR	Byte not programmed successfully.
	LOW BAT ERR	The HP-41 battery voltage dropped too low to continue reliable operation.
	TRANSMIT ERR	Loop not operating correctly.

GETEP

FUNCTION: Reads the contents of an L8 and a U2 EPROM into a 4k block of RAM. The EPROMs are read from the first MC00506A on the loop starting with the primary device. The user is first prompted to "RDY L8 EPROM". When the EPROM (and its personality module) are ready, any key may be pressed to start the read. The "ON" key may be pressed instead to halt the program at this point. When the L8 EPROM has been read, the user will be prompted with "RDY U2 EPROM". When the U2 EPROM is ready, press any key to start the read. As before, the "ON" key will stop the program.

INPUT: Z REG - The section number to read of the L8 EPROM (0 - 1).
Y REG - The section number to read of the U2 EPROM (0 - 3).
X REG - The block number to read into (0 - 15).

OUTPUT: NONE

ERRORS:		
	DATA ERROR	Block number (X) > 15 or L8 section number (Z) > 1 or U2 section number (Y) > 3.
	ALPHA DATA	Block number or section number is type ALPHA rather than NUMERIC.

NO HPIL	HPIL module is not plugged in.
NO MC00506A	EPROM Programmer not on loop or loop in MANIO and MC00506A not the primary device.
TRANSMIT ERR	Loop not operating correctly.

*** MASS STORAGE FUNCTIONS ***

BACKGROUND ON HP-41 ROM STORAGE

The programs "ROM)REG" and "REG)ROM" were developed and published in the PPC Calculator Journal (V9N3P40) to facilitate the transfer of ROM programs between HP-41 users. They allow ROM words to be transferred to and from the data registers of the HP-41, so that any mass storage device may be used to permanently store them.

Since HP-41 object code is not position independent, the programs use a header code that allows ROM words to be easily loaded back at the address from which they were saved. This allows HP-41 assembly language programmers to exchange single routines on magnetic cards, for instance.

The most common use of ROM)REG and REG)ROM is to save an entire 4k block on cassette tape. Because 4k of ROM code will not fit into the HP-41 data registers all at once, the 4k block has to be 'ROM)REGed' in several smaller pieces, which are each written to tape as DATA files.

Since this process is rather slow, and requires the use of almost all the HP-41 memory as a temporary buffer, this EPROM set includes two new routines: READROM and WRTROM. These routines read and write directly from an MLDL type device to the HPIL mass storage devices, and do not use any memory within the HP-41. They are fully compatible with the 'ROM)REG' standard, however. They store a ROM in four separate 1k blocks within an 824 register DATA file.

ROM)REG
REG)ROM

See the original ROM)REG article in PPC Calculator Journal V9N3P40.

READROM

FUNCTION: Load a 'ROM)REG' format ROM image file from an HP-IL mass storage device into a block of MLDL RAM. If the loop is in AUTOIO mode, all mass storage devices on the loop will be searched (starting with the primary device) to find the file. If the loop is in MANID, only the current primary device will be used. The specified block is not checked to make sure that it is really RAM -- this is up to the user.

INPUT: X REG - The block number (0 - 15) to load into.
ALPHA - Filename of the ROM image file.

OUPUT: NONE

ERRORS:		
	DATA ERROR	Block number) 15.
	ALPHA DATA	Block number is type ALPHA rather than NUMERIC.
	NO HPIL	HPIL module is not plugged in.
	NO DRIVE	An HPIL mass storage device is not on the loop.
	NO MEDM	Medium is not installed properly.
	MEDM ERR	Medium improperly installed, worn out, or damaged.
	DRIVE ERR	Medium stalled or bad drive.
	NAME ERR	ALPHA register is empty.
	FL NOT FOUND	Named file is not on the medium.
	FL TYPE ERR	Specified file is not a DATA file.
	FORMAT ERR	The specified file is not in the 'ROM)REG' format or is not exactly 824 registers long.
	TRANSMIT ERR	Loop not operating correctly.

WRTROM

FUNCTION: Store the contents of an HP-41 4k ROM block to a mass storage file in the 'ROM)REG' format. If the named file exists and is of the correct type and length, it is overwritten with the new ROM image (unless the file is SECURED). If the file does not exist, it is automatically created.

INPUT: X REG - The block number (0 - 15) to write.
ALPHA - The filename of the new file.

OUTPUT: NONE

ERRORS:		
	DATA ERROR	Block number) 15.
	ALPHA DATA	Block number is type ALPHA rather than NUMERIC.
	NO HPIL	HPIL module is not plugged in.
	NO DRIVE	An HPIL mass storage device is not on the loop.
	NO MEDM	Medium is not installed properly.
	MEDM ERR	Medium improperly installed, worn out, or damaged.
	DRIVE ERR	Medium stalled or bad drive.
	NAME ERR	ALPHA register is empty.
	DUP FL NAME	The specified file already exists on the medium, but is not an 824 register DATA file (which would be overwritten).
	FL SECURED	A file of the specified name and of the correct size and type exists on the medium, but cannot be overwritten because it is secured.
	DIR FULL	The file cannot be created because the directory is full.
	MEDM FULL	The file cannot be created because the medium does not have enough storage space left.

TRANSMIT ERR Loop not operating correctly.

*** OTHER FUNCTIONS ***

RCOPY

FUNCTION: Copy any 4k ROM into a block of RAM. The destination block is not checked to assure that it is really RAM, this is up to the user.

INPUT: Y REG – The block number (0 – 15) of the source ROM.
X REG – The block number (0 – 15) of the destination ROM (RAM).

OUTPUT: NONE

ERRORS: DATA ERROR Either block number (X or Y)) 15.
ALPHA DATA Either block number (X or Y) ALPHA DATA.

PCAT

FUNCTION: Displays a CAT 2 listing starting with a particular port number. Numbers 1 through 4 start the CAT listing with that port (ROM addresses 8000, A000, C000, and E000, respectively) and continue through all modules in higher numbered ports. Numbers 5, 6, or 7 start the Catalog at ROM addresses 5000, 6000 , or 7000 which correspond to the addresses of the Timer, Printer, and HPIL modules.

INPUT: Single digit 1-7 at prompt.

OUTPUT: NONE

ERRORS: NONE

KASN

FUNCTION: Makes 2 byte key assignments in a similar manner to "1K" of the PPC ROM. Differences are that an assignment will always overwrite any existing assignment to a key, and that if both assignment bytes are zero, the key will have any present key assignment cleared.

INPUT: Z REG – First byte of assignment (0-255)
Y REG – Second byte of assignment (0-255)
X REG – Keycode (Standard row/column)

OUTPUT: NONE

ERRORS: DATA ERROR Either assignment byte greater than 255 or invalid keycode.

PACKING, TRY AGAIN Not enough room to make assignment.

E P R O M L I S T I N G

```
Puget Sound Programing HP-41 Assembler: mceprom.a
00001   0000        start     equ                   $A000
00002   A000                  org     start
00003
00004   A000 00F              con     15            XROM #
00005   A001 00B              con     11            # of functions
00006
00007   A002 000              fat     romname
00007   A003 099
00008   A004 000              fat     burnl
00008   A005 09F
00009   A006 000              fat     burnu
00009   A007 0A6
00010   A008 002              fat     getep
00010   A009 0F6
00011   A00A 003              fat     kasn
00011   A00B 0B9
00012   A00C 004              fat     pcat
00012   A00D 00C
00013   A00E 001              fat     rcopy
00013   A00F 051
00014   A010 001              fat     readrom
00014   A011 06D
00015   A012 001              fat     regrom
00015   A013 0AF
00016   A014 002              fat     romreg
00016   A015 096
00017   A016 002              fat     wrtrom
00017   A017 003
00018
00019   A018 000              con     0
00020   A019 000              con     0
00021
00022   A090                  org     $A090
00023
00024                 ************************************************************
00025                 * ROM name                                                 *
00026                 ************************************************************
00027   A090 08D              con     $8D
00028   A091 00F              con     $0F
00029   A092 012              con     $12
00030   A093 010              con     $10
00031   A094 005              con     $05
00032   A095 020              con     $20
00033   A096 003              con     $03
00034   A097 00D              con     $0D
00035   A098 02D              con     $2D           name: -MC EPROM
00036   A099 3E0    romname   rtn
00037
00038
00039
00040                 ************************************************************
00041                 * BURNL - Burn an LB EPROM image on the MC00506A           *
00042                 *       - INPUT: X - The HP41 ROM block to burn (0-15)     *
00043                 *       -        Y - The 4k section of the LB to burn      *
00044                 *                    (used for upper/lower half of 2764)   *
00045                 ************************************************************
00046
00047   A09A 08C              con     $8C
00048   A09B 00E              con     $0E
00049   A09C 012              con     $12
00050   A09D 015              con     $15
```


PPC

```
00051   A09E 002            con     $02             name: BURNL
00052
00053   A09F 3C4    burnl   st=0
00054   A0A0 043            jnc     burnlu10
00055
00056
00057                       ************************************************************
00058                       * BURNU - Burn a U2 EPROM on the MC00506A                   *
00059                       *       - INPUT: X - The HP41 ROM block to burn (0-15)      *
00060                       *       -        Y - The 1k EPROM section to burn (0-4)     *
00061                       ************************************************************
00062
00063   A0A1 095            con     $95
00064   A0A2 00E            con     $0E
00065   A0A3 012            con     $12
00066   A0A4 015            con     $15
00067   A0A5 002            con     $02             Name: BURNU
00068
00069   A0A6 3C4    burnu   st=0                    clear flags
00070   A0A7 208            setf    2               indicates U2 burn
00071
00072                       * Display function name, do PILTST, and find the MC00506
00073   A0A8 10E    burnlu10 a=c    all             move entry address to A
00074   A0A9 36D            xqrel   dispname        display name with ':'
00074   A0AA 0BC
00074   A0AB 0A0
00075   A0AC 36D            xqrel   findmc          PIL test and MC00506 find
00075   A0AD 0BC
00075   A0AE 126
00076
00077                       * Use the block number in X to make address and counter
00078   A0AF 36D            xqrel   bcdbinXF        X -> bin, max value is 15
00078   A0B0 0BC
00078   A0B1 0E8
00079   A0B2 13C            rcr     8               left 6, make it an adr
00080   A0B3 266            c=c-1   s&x             S&X = FFF (ctr for L8)
00081   A0B4 20C            ?fset   2               doing U2 ?
00082   A0B5 01B            jnc     burnlu20        ..no, skip
00083   A0B6 130            ldi     $3FF            S&X = 3FF (ctr for U2)
00083   A0B7 3FF
00084   A0B8 070    burnlu20 n=c                    save ctr/adr in N
00085
00086                       * Use section number in Y to make start & end EPROM address
00087   A0B9 130            ldi     2               1 is max section # for L8
00087   A0BA 002
00088   A0BB 20C            ?fset   2               doing U2 ?
00089   A0BC 01B            jnc     burnlu30        ..no, skip
00090   A0BD 130            ldi     4               3 is max section # for U2
00090   A0BE 004
00091   A0BF 0E6    burnlu30 c()b   s&x             max value to B
00092   A0C0 0B8            read    2               get the Y register
00093   A0C1 10E            a=c     all             put it in A
00094   A0C2 36D            xqrel   bcdbin          convert to binary
00094   A0C3 0BC
00094   A0C4 0ED
00095
00096   A0C5 20C            ?fset   2               doing U2 ?
00097   A0C6 037            jc      burnlu40        ..yes, skip the L8 part
00098   A0C7 1BC            rcr     11              left 3 - adr is 0000 or 1000
00099   A0C8 158            m=c                     start adr to M
00100   A0C9 266            c=c-1   s&x             end adr (xFFF)
00101   A0CA 10E            a=c     all             end adr to A
00102   A0CB 04B            jnc     burnlu50        skip past U2 stuff
00103
```

```
00104   A0CC 1E6   burnlu40  c=c+c   s&x
00105   A0CD 1E6             c=c+c   s&x            0,1,2,3 -> 0,4,8,C
00106   A0CE 37C             rcr     12             left 2, adr is 0000 to 0C00
00107   A0CF 158             m=c                    start adr in M
00108   A0D0 10E             a=c     all            adr to A
00109   A0D1 130             ldi     $3FF
00109   A0D2 3FF
00110   A0D3 146             a=a+c   s&x            end = start + 3FF
00111
00112                * Initialize the MC00506 - start adr in M, end adr in A
00113                *      Does init, blank check, resets start adr
00114   A0D4 36D   burnlu50  xqrel   initmce
00114   A0D5 08C
00114   A0D6 037
00115
00116                * Do first fetch, then start burning loop
00117   A0D7 349             xqrel   prefetch       get 1st byte of ROM
00117   A0D8 08C
00117   A0D9 0EA
00118   A0DA 308             setf    1              indicate 2nd to Nth bytes
00119
00120
00121                ********************************************************
00122                * This is the main programming loop                     *
00123                ********************************************************
00124   A0DB 375   burnlu    ?ncxq   $70DD          LAD to MC00506
00124   A0DC 1C0
00125   A0DD 0CE             c=b     all            get the byte from B
00126   A0DE 099             ?ncxq   $7126          send 1st DAB
00126   A0DF 1C4
00127
00128                * Source a GET to burn the byte
00129   A0E0 064             selp    4              IL reg 1
00130   A0E1 205             con     $205           init R1 for CMD
00131   A0E2 0A4             selp    5              IL reg 2
00132   A0E3 021             con     $021           source the GET (hex08)
00133
00134                * Display the current address and increment it
00135   A0E4 36D             xqrel   adr2dis
00135   A0E5 08C
00135   A0E6 0AF
00136   A0E7 198             c=m
00137   A0E8 22E             c=c+1   all
00138   A0E9 158             m=c
00139
00140                * Flag 2 determines L8 or U2
00141   A0EA 20C   prefetch  ?fset   2
00142   A0EB 04F             jc      u2fetch
00143
00144                * Fetch the next L8 ROM word
00145   A0EC 090             c=n                    get the adr/ctr
00146   A0ED 0E6             c()b    s&x            save ctr in B
00147   A0EE 330             fetch                  get the word
00148
00149   A0EF 0E6             c()b    s&x            ROM word to B, ctr to C
00150   A0F0 23A             c=c+1   m              incr address
00151   A0F1 070             n=c                    save adr & ctr in N
00152   A0F2 0A3             jnc     burn10         bypass u2fetch
00153
00154   A0F3 343   brnlujmp  jnc     burnlu         stepping stone
00155
00156
```

```
00157                   * Fetch the next U2 ROM word
00158   A0F4 0B0  u2fetch   c=n                 get adr to C
00159   A0F5 0E6            c()b    s&x         save ctr in B
00160   A0F6 006            a=0     s&x         clr bit ACC
00161   A0F7 05C            r=      4           set loop ctr
00162
00163   A0F8 330  u2fet10   fetch               get ROM word
00164   A0F9 23A            c=c+1   m           incr ROM adr
00165   A0FA 116            a=c     xs          u2 bits to A
00166   A0FB 0AE            a()c    all
00167   A0FC 1E6            c=c+c   s&x
00168   A0FD 1E6            c=c+c   s&x         push bits to upper end of s&x
00169   A0FE 3C6            rshfc   s&x         put u2 bits in C d1,0
00170   A0FF 0AE            a()c    all
00171   A100 3D4            r=r-1               decr loop ctr
00172   A101 394            ?r=     0           end of loop
00173   A102 3B3            jnc     u2fet10     ..no, loop
00174
00175   A103 0C6            c=b     s&x         ctr to C, with ROM adr
00176   A104 070            n=c                 save ctr/adr
00177   A105 086            b=a     s&x         save u2 byte in B
00178
00179   A106 30C  burn10    ?fset   1           F1 clear for initialization
00180   A107 3A0            ?ncrtn              return if init call
00181
00182
00183                   * error check the GET frame
00184   A108 2F5            ?ncxq   $70BD
00184   A109 1C0
00185
00186                   * TAD to MC00506
00187   A10A 2C9            ?ncxq   $70B2       TAD r5
00187   A10B 1C0
00188
00189                   * Send the SST and get the DAB
00190   A10C 130            ldi     $061
00190   A10D 061
00191   A10E 0D9            ?ncxq   $7136
00191   A10F 1C4
00192
00193                   * Do a TRANSMIT ERROR check
00194   A110 39D            ?ncxq   $77E7
00194   A111 1DC
00195
00196                   * The status byte was returned in digits 1 & 2 of A.
00197   A112 386            rshfa   s&x         status to A d0,1
00198   A113 0A6            a()c    s&X         status to C
00199   A114 3D8            c()st               status into flags
00200   A115 38C            ?fset   0
00201   A116 0E7            jc      burnerr     BAD BURN
00202   A117 160            ?lowbat
00203   A118 157            jc      lowbaterr   LOW BATTERY HALT
00204   A119 3D8            c()st               put back the flags
00205   A11A 3C8            clrkey
00206   A11B 3CC            ?key                key down ?
00207   A11C 02B            jnc     loopctrl    ..no, continue
00208   A11D 220            c=key               get the key
00209   A11E 01C            r=      3           pt to LS key digit
00210   A11F 1E2            c=c+c   @r          gen carry if key was 'ON'
00211   A120 037            jc      burnexit    quit if key was 'ON'
00212
00213                   * Looping control
00214   A121 0B0  loopctrl  c=n                 get adr, ctr
00215   A122 266            c=c-1   s&x
```

```
00216   A123 01F              jc      burnexit
00217   A124 070              n=c                     restore new ctr
00218   A125 273              jnc     brnlujmp        do another loop
00219
00220   A126 3C1    burnexit  ?ncxq    $2CF0           clear the display
00220   A127 0B0
00221   A128 149              ?ncxq    $0952           deselect display & rtn
00221   A129 024
00222   A12A 375              ?ncxq    $70DD           LAD to MC00505
00222   A12B 1C0
00223   A12C 130              ldi      $004            SDC - reset the MC00505
00223   A12D 004
00224   A12E 2E9              ?ncxq    $70BA           CMD frame
00224   A12F 1C0
00225   A130 0D9              ?ncgo    $7336           UNL, error check, & RTN
00225   A131 1CE
00226
00227
00228                 * Bad Burn error message
00229   A132 2CD    burnerr   ?ncxq    $77B3           Put mssg in dsp
00229   A133 1DC
00230   A134 002              con      $02
00231   A135 015              con      $15
00232   A136 012              con      $12
00233   A137 20E              con      $20E            mssg: BURN
00234   A138 015    brnERRmsg ?ncxq    $7D05           Add ERR and do error
00234   A139 1F4
00235
00236                 * Erase Error mssg
00237   A13A 2CD    eraserr   ?ncxq    $77B3           put mssg in dsp
00237   A13B 1DC
00238   A13C 005              con      $05
00239   A13D 012              con      $12
00240   A13E 001              con      $01
00241   A13F 013              con      $13
00242   A140 205              con      $205            mssg: ERASE
00243   A141 3BB              jnc      brnERRmsg
00244
00245                 * Low battery error
00246   A142 2CD    lowbaterr ?ncxq    $77B3           put mssg in dsp
00246   A143 1DC
00247   A144 00C              con      $0C
00248   A145 00F              con      $0F
00249   A146 017              con      $17
00250   A147 020              con      $20
00251   A148 002              con      $02
00252   A149 001              con      $01
00253   A14A 214              con      $214            mssg: LOW BAT
00254   A14B 3BB              jnc      brnERRmsg
00255
00256                 ************************************************************
00257                 *RCOPY - Function to copy 4k blocks.                       *
00258                 *       - INPUT: Y - Source block.                         *
00259                 *       -        X - Destination block.                    *
00260                 * No error checking is done to insure that the destination  *
00261                 * block is really RAM. This is up to the user.              *
00262                 ************************************************************
00263
00264   A14C 099              con      $99
00265   A14D 010              con      $10
00266   A14E 00F              con      $0F
00267   A14F 003              con      $03
00268   A150 012              con      $12             name: RCOPY
```

```
00269
00270   A151 36D   rcopy     xqrel   bcdbinXF      X reg (dest #) range 0-15
00270   A152 08C
00270   A153 0E8
00271   A154 13C            rcr     8             left 6 to make adr
00272   A155 0FA            c()b    m             put dest adr in B Mant
00273   A156 0B8            read    2             get src blk #
00274   A157 10E            a=c     all
00275   A158 36D            xqrel   bcdbin        convert to bin, range 0-15
00275   A159 08C
00275   A15A 0ED
00276   A15B 13C            rcr     8             left 6 to make adr
00277   A15C 00E            a=0     all           init the counter
00278   A15D 330   rcopy10   fetch                 get the src word
00279   A15E 23A            c=c+1   m             incr src adr
00280   A15F 0FA            c()b    m             get dest adr
00281   A160 040            writr                 write the dest word
00282   A161 23A            c=c+1   m             incr dest adr
00283   A162 0FA            c()b    m             src adr back to C
00284   A163 166            a=a+1   s&x           incr ctr
00285   A164 3CB            jnc     rcopy10
00286   A165 3E0            rtn
00287
00288                *****************************************************************
00289                * READROM - Read a ROM)REG format tape file into emulated ROM   *
00290                *           - INPUT: Filename in ALPHA                          *
00291                *                    Block number in X                          *
00292                *           - OUTPUT: 4K block loaded to EROM                   *
00293                *****************************************************************
00294
00295   A166 08D            con     $8D
00296   A167 00F            con     $0F
00297   A168 012            con     $12
00298   A169 004            con     $04
00299   A16A 001            con     $01
00300   A16B 005            con     $05
00301   A16C 012            con     $12           name: READROM
00302
00303   A16D 36D   readrom   xqrel   piltst        test for existence of PIL
00303   A16E 08C
00303   A16F 176
00304   A170 36D            xqrel   bcdbinXF      X reg (blk #) in range 0-15
00304   A171 08C
00304   A172 0E8
00305   A173 2B8            read    10            status scratch
00306   A174 0A6            a()c    s&x           put in block number
00307   A175 2A8            writ    10            save block # in Status 10
00308
00309   A176 021            ?ncxq   $7808         get file params from DIR
00309   A177 1E0
00310   A178 130            ldi     824           exp. file size
00310   A179 338
00311   A17A 106            a=c     s&x
00312   A17B 0B0            c=n                   get file params
00313   A17C 366            ?a#c    s&x           is filesize () 824 regs
00314   A17D 05B            jnc     rdrom20       ..ok, continue
00315
00316   A17E 2CD   fmterr    ?ncxq   $77B3         put mssg in display
00316   A17F 1DC
00317   A180 006            con     $06
00318   A181 00F            con     $0F
00319   A182 012            con     $12
00320   A183 00D            con     $0D
00321   A184 001            con     $01
```

```
00322   A185 214              con     $214            mesg: FORMAT
00323   A186 015              ?ncxq   $7D05           Add ERR and do error
00323   A187 1F4
00324
00325
00326   A188 1DD    rdrom20   ?ncxq   $7F77           Seek the file
00326   A189 1FC
00327   A18A 399              ?ncxq   $70E6           Send buffer 0 and SDA
00327   A18B 1C0
00328   A18C 0E0              slctq
00329   A18D 05C              r=      4               loop counter
00330   A18E 104              clrf    8               indicate reg fetch from PIL
00331
00332   A18F 0A0    rdrom30   slctp
00333   A190 0F9              ?ncxq   $763E           get the header reg
00333   A191 1DB
00334   A192 070              n=c                     header to N
00335   A193 1BC              rcr     11              left 3
00336   A194 280              ilw     2               retransmit the last DAB
00337   A195 130              ldi     $3FF            header count must be 3FF
00337   A196 3FF
00338   A197 366              ?a#c    s&x             count () 3FF ?
00339   A198 337              jc      fmterr          ..yes, format error
00340   A199 1BE              a=a-1   ms
00341   A19A 1BE              a=a-1   ms              check MS = 1 ?
00342   A19B 31B              jnc     fmterr
00343   A19C 2B8              read    10              get status 10, block # in bin.
00344   A19D 05A              c=0     m
00345   A19E 158              m=c                     block # in M
00346   A19F 349              xqrel   regrom10        1024 bytes to EROM
00346   A1A0 08C
00346   A1A1 1BD
00347   A1A2 0E0              slctq
00348   A1A3 3D4              r=r-1                   decr loop ctr
00349   A1A4 394              ?r=     0               end of loop ?
00350   A1A5 353              jnc     rdrom30         ..no, continue
00351
00352   A1A6 2B1              ?ncgo   $70AC           UNT and rtn
00352   A1A7 1C2
00353
00354
00355                 ********************************************************
00356                 *REG)ROM                                                *
00357                 ********************************************************
00358
00359   A1A8 08D              con     "8D"
00360   A1A9 00F              con     "0F"
00361   A1AA 012              con     "12"
00362   A1AB 03E              con     "3E"
00363   A1AC 007              con     "07"
00364   A1AD 005              con     "05"
00365   A1AE 012              con     "12"            name: REG)ROM
00366
00367   A1AF 0B8    regrom    read    2               get the start adr from Y
00368   A1B0 158              m=c                     save in M
00369   A1B1 0F8              read    3               get start reg #
00370   A1B2 38D              ?ncxq   $02E3           convert to binary
00370   A1B3 008
00371   A1B4 106              a=c     s&x             move reg # to A
00372   A1B5 378              read    13
00373   A1B6 03C              rcr     3               move curtain to s&x
00374   A1B7 206              c=c+a   s&x             compute reg adr
00375   A1B8 270              ramslct
00376   A1B9 0E6              c()b    s&x             save reg adr in B
00377   A1BA 038              read    0               get the header
```


PPC

```
00378   A1BB 070            n=c                 save in N
00379   A1BC 108            setf   8            indicate REG)ROM
00380
00381                   * Entry point for READROM
00382   A1BD 0B0    regrom10  c=n                 get header from N
00383   A1BE 03C              rcr    3            AAAA to LS 4 digits
00384   A1BF 10E              a=c    all          move to A
00385   A1C0 198              c=m                 get the start adr
00386   A1C1 33C              rcr    1            split start adr across MS,S&X
00387   A1C2 05A              c=0    m            clear rest of digits
00388   A1C3 2E6              ?c#0   s&x          test upper 3 digits
00389   A1C4 047              jc     regrom20     ) 000F - abs address
00390   A1C5 0FC              rcr    10           (= ? , move 4 left
00391   A1C6 0A6              a()c   s&x          move lowr 3 of AAAA to C
00392   A1C7 01C              r=     3            pt to MS address digit
00393   A1C8 2E2              ?c#0   @r           test that digit
00394   A1C9 027              jc     regrom30     if () 0, it's 4k blk adr
00395   A1CA 0A2              a()c   @r           else same 4k blk as before
00396   A1CB 013              jnc    regrom30     unconditional
00397   A1CC 2FC    regrom20  rcr    13
00398   A1CD 1BC    regrom30  rcr    11           left 3
00399   A1CE 158              m=c                 save current ROM adr in M
00400   A1CF 10E              a=c    all          and in A
00401   A1D0 0B0              c=n                 get the header again
00402   A1D1 05A              c=0    m            clr upper digits, NNN in s&x
00403   A1D2 1BC              rcr    11           left 3 to adr field in Mant
00404   A1D3 15A              a=a+c  m            add NNN to strt to get final
00405   A1D4 09A              b=a    m            final ROM adr in B Mantissa
00406
00407
00408
00409                   * Main loop for REG)ROM
00410   A1D5 10C    regrom40  ?fset  8            src from regs or IL ?
00411   A1D6 037              jc     regrom50     get src from Regs
00412
00413                   * Fetch register from HPIL (Cassette)
00414   A1D7 0F9              ?ncxq  $763E        fetch next tape register
00414   A1D8 1D8
00415   A1D9 1BC              rcr    11           left 3
00416   A1DA 280              ilw    2            retransmit last DAB
00417   A1DB 03B              jnc    regrom60     continue...
00418
00419                   * Fetch from data registers
00420   A1DC 0C6    regrom50  c=b    s&x          get old reg adr
00421   A1DD 226              c=c+1  s&x          incr to current reg
00422   A1DE 270              ramslct             slct the reg
00423   A1DF 0E6              c()b   s&x          save the new reg adr
00424   A1E0 038              read   0            get the register
00425   A1E1 10E              a=c    all          move register to A
00426
00427                   * Resume common processing with register in A
00428   A1E2 198    regrom60  c=m                 get current ROM adr
00429   A1E3 0AE              a()c   all          ROM adr to A, register to C
00430   A1E4 2FC              rcr    13
00431   A1E5 1EE              c=c+c  all
00432   A1E6 1EE              c=c+c  all          left justify bits in C
00433   A1E7 09C              r=     5            set the inner loop counter
00434
00435   A1E8 37C    regrom70  rcr    12           rotate C left 8 bits
00436   A1E9 056              c=0    xs           clear upper bits of word
00437   A1EA 1EE              c=c+c  all
00438   A1EB 013              jnc    *+02
00439   A1EC 22E              c=c+1  all
00440   A1ED 1EE              c=c+c  all
00441   A1EE 013              jnc    *+02         rotate C left 2 more bits
```

```
00442   A1EF 22E            c=c+1   all         end with ROM word in s&x
00443   A1F0 106            a=c     s&x         move ROM word to A
00444   A1F1 0AE            a()c    all         string to A, Adr-word to C
00445   A1F2 040            writr               write to ROM
00446   A1F3 0AE            a()c    all         adr-word to A, string to C
00447   A1F4 33A            ?a(b    m           current adr ( final adr
00448   A1F5 3A0            ?ncrtn              ..no, stop
00449   A1F6 17A            a=a+1   m           incr the address
00450   A1F7 3D4            r=r-1               decr ctr
00451   A1F8 394            ?r=     0           end of inner loop?
00452   A1F9 37B            jnc     regrom70    ..no, continue inner loop
00453
00454   A1FA 0AE            a()c    all         ROM adr to C
00455   A1FB 158            m=c                 ROM adr to M
00456   A1FC 2CB            jnc     regrom40    MAIN Loop
00457
00458                *****************************************************
00459                *WRTROM - Copy a ROM image (in ROM)REG format) to a tape file*
00460                *       - The file is created if neccessary.          *
00461                *       - INPUT: Filename in ALPHA                    *
00462                *                Block number to save in X            *
00463                *       - OUTPUT: Block saved                         *
00464                *****************************************************
00465
00466   A1FD 08D            con     $8D
00467   A1FE 00F            con     $0F
00468   A1FF 012            con     $12
00469   A200 014            con     $14
00470   A201 012            con     $12
00471   A202 017            con     $17         name: WRTROM
00472
00473   A203 36D  wrtrom    xqrel   piltst      test for HPIL existence
00473   A204 08C
00473   A205 176
00474   A206 36D            xqrel   bcdbinXF    test X reg for range 0-15
00474   A207 08C
00474   A208 0E8
00475
00476                * Check for named file existent
00477   A209 05E            c=0     ms
00478   A20A 02D            ?ncxq   $780B       ck any type, same name
00478   A20B 1E0
00479   A20C 198            c=m                 M = 0 if no file
00480   A20D 2EE            ?c#0    all         did file exist ?
00481   A20E 08B            jnc     newfile     ..no, create it
00482
00483                * Named file exists: must be DATA, 824 Regs, Unsecured
00484   A20F 0B0            c=n                 get the file pointer
00485   A210 10E            a=c     all         FP to A
00486   A211 130            ldi     $D
00486   A212 00D
00487   A213 33C            rcr     1
00488   A214 37E            ?a#c    ms          is the file DATA, type D
00489   A215 01B            jnc     wrtrom10    ..yes, continue
00490   A216 249  duperr    ?ncgo   $7692       DUP FL NAME error
00490   A217 1DA
00491   A218 130  wrtrom10  ldi     824
00491   A219 338
00492   A21A 366            ?a#c    s&x         size = 824 regs ?
00493   A21B 3DF            jc      duperr      ..no, DUP FL NAME
00494   A21C 031            ?ncxq   $7D0C       test secured
00494   A21D 1F4
```

```
00495   A21E 06B            jnc     wrtrom30        file OK, use it
00496
00497                   * Create a new file - type DATA, 824 regs, 26 records
00498   A21F 04E    newfile c=0     all             ready to make new file ptr
00499   A220 130            ldi     $D0             type and status
00499   A221 0D0
00500   A222 23C            rcr     2
00501   A223 130            ldi     824             # of regs
00501   A224 338
00502   A225 07C            rcr     4
00503   A226 130            ldi     26              # of records
00503   A227 01A
00504   A228 07C            rcr     4
00505   A229 2F1            ?ncxq   $76BC           make the file
00505   A22A 1D8
00506
00507                   * File pointer is in N, get ready to write the file
00508   A22B 1C9    wrtrom30 ?ncxq  $7F72           seek to file, set BP = 0
00508   A22C 1FC
00509   A22D 130            ldi     $A2             DDL 2 - WRITE mode
00509   A22E 0A2
00510   A22F 2E9            ?ncxq   $70BA           CMD
00510   A230 1C0
00511   A231 369            ?ncxq   $70DA           DDL 0 - Write buffer 0
00511   A232 1C0
00512   A233 39D            ?ncxq   $77E7           error check
00512   A234 1DC
00513   A235 064            selp    4
00514   A236 005            con     $005            set r1 for DAB
00515
00516
00517
00518                   * Make first header register
00519   A237 36D            xqrel   bcdbinXF        X reg (blk #) to binary (0-15)
00519   A238 08C
00519   A239 0E8
00520   A23A 33C            rcr     1               block # to ms
00521   A23B 05A            c=0     m
00522   A23C 130            ldi     206             # of regs in file
00522   A23D 0CE
00523   A23E 2BC            rcr     7
00524   A23F 130            ldi     $3FF            byte count
00524   A240 3FF
00525   A241 2DC            r=      13
00526   A242 0D0            ld0r    3               loop ctr in MS of header
00527
00528   A243 158    wrtrom50 m=c                    save new header in M
00529   A244 05E            c=0     ms
00530   A245 23E            c=c+1   ms              type header as ALPHA
00531   A246 33D            ?ncxq   $75CF           send the register
00531   A247 1D4
00532   A248 39D            ?ncxq   $77E7           error check
00532   A249 1DC
00533   A24A 198            c=m                     get the header back
00534   A24B 00E            a=0     all
00535   A24C 106            a=c     s&k             copy the 3FF to A
00536   A24D 07C            rcr     4
00537   A24E 05A            c=0     m               clear header except st adr
00538   A24F 2FC            rcr     13              adr in C d3-0
00539   A250 14E            a=a+c   all             end adr in A
00540   A251 3EE            lshfa   all
00541   A252 3EE            lshfa   all
00542   A253 3EE            lshfa   all             end adr to d6-3 of A
00543   A254 1BC            rcr     11              st adr to d6-3 of C
00544   A255 0EE            c()b    all             st adr to B
```

```
00545   A256 108            setf   8
00546   A257 349            xqrel  rmrgguts      ROM)REG the 1k block
00546   A258 08C
00546   A259 260
00547   A25A 198            c=m                  get the header
00548   A25B 09C            r=     5             pt to A2 digit (of A3-0)
00549   A25C 222            c=c+1  @r
00550   A25D 222            c=c+1  @r
00551   A25E 222            c=c+1  @r
00552   A25F 222            c=c+1  @r            add $400 to start adr
00553   A260 27E            c=c-1  ms            decrement loop ctr
00554   A261 313            jnc    wrtrom50      next 1k block
00555
00556                * Close the record and clean up
00557   A262 130            ldi    $A8           DDL 8
00557   A263 0A8
00558   A264 2E9            ?ncxq  $70BA         CMD
00558   A265 1C0
00559   A266 2BD            ?ncgo  $70AF         UNL
00559   A267 1C2
00560
00561
00562                * This is the guts of ROM)REG and WRTROM
00563
00564   A268 06E  rmrgguts  a()b   all
00565   A269 09C            r=     5
00566   A26A 0AE  rmrg10    a()c   all           adr to C, str to A
00567   A26B 3EE            lshfa  all
00568   A26C 3EE            lshfa  all           A 2 digits left
00569   A26D 330            fetch
00570   A26E 106            a=c    s&x           ROM word to A
00571   A26F 0AE            a()c   all           str to C, adr to A
00572   A270 1E6            c=c+c  s&x
00573   A271 1E6            c=c+c  s&x           push word against end of str
00574   A272 1EE            c=c+c  all
00575   A273 1EE            c=c+c  all           shift str to even digit
00576   A274 17A            a=a+1  m             incr adr
00577   A275 3D4            r=r-1
00578   A276 394            ?r=    0             end of loop ?
00579   A277 39B            jnc    rmrg10        ..no, continue looping
00580   A278 3CE            rshfc  all           right justify str in C
00581   A279 23E            c=c+1  ms            type as ALPHA
00582   A27A 10C            ?fset  8             ROM)REG or WRTROM ?
00583   A27B 03F            jc     rmrg20        ..do WRTROM
00584
00585   A27C 1D8            c()m                 str to M/reg adr to C
00586   A27D 226            c=c+1  s&x           incr reg adr
00587   A27E 270            ramslct
00588   A27F 1D8            c()m                 str to C/reg adr to M
00589   A280 2F0            writd                write the register
00590   A281 053            jnc    rmrg30        skip past WRTROM section
00591
00592   A282 0AE  rmrg20    a()c   all           str to A, adr to C
00593   A283 070            n=c                  save adr in N
00594   A284 0AE            a()c   all           str to C
00595   A285 33D            ?ncxq  $75CF         send to PIL
00595   A286 1D4
00596   A287 39D            ?ncxq  $77E7         error check
00596   A288 1DC
00597   A289 0B0            c=n                  get adr back
00598   A28A 10E            a=c    all           adr to A
00599
00600   A28B 06E  rmrg30    a()b   all           swap start/end adr
00601   A28C 33A            ?a(b   m             EEEE ( BBBB ?
00602   A28D 2DB            jnc    rmrgguts
```


PPC

```
00603   A28E 3E0            rtn
00604
00605
00606
00607               ********************************************************************
00608               * ROM)REG - Pack ROM words into HP-41 registers.                   *
00609               *         - INPUT: X - Starting register number.                   *
00610               *                  Y - Binary start/end ROM adrs (BBBBEEEE).       *
00611               *         - OUTPUT: L - The number of the last register used.      *
00612               * See PPCCJ V9N3P41 for source code comments.                       *
00613               ********************************************************************
00614
00615   A28F 087            con     $87
00616   A290 005            con     $05
00617   A291 012            con     $12
00618   A292 03E            con     $3E
00619   A293 00D            con     $0D
00620   A294 00F            con     $0F
00621   A295 012            con     $12             name: ROM)REG
00622
00623   A296 349    romreg  xqrel   lastreg
00623   A297 0BC
00623   A298 2DD
00624   A299 046            c=0     s&x
00625   A29A 270            ramslct
00626   A29B 0F8            read    3
00627   A29C 38D            ?nckq   $02E3
00627   A29D 008
00628   A29E 106            a=c     s&x
00629   A29F 378            read    13
00630   A2A0 03C            rcr     3
00631   A2A1 206            c=c+a   s&x
00632   A2A2 150            m=c
00633   A2A3 106            a=c     s&x
00634   A2A4 066            a()b    s&x
00635   A2A5 246            c=a-c   s&x
00636   A2A6 106            a=c     s&x
00637   A2A7 1E6            c=c+c   s&x
00638   A2A8 1E6            c=c+c   s&x
00639   A2A9 146            a=a+c   s&x
00640   A2AA 086            b=a     s&x
00641   A2AB 098            read    2
00642   A2AC 00E            a=0     all
00643   A2AD 01C            r=      3
00644   A2AE 10A            a=c     r(-
00645   A2AF 0BC            rcr     5
00646   A2B0 05A            c=0     m
00647   A2B1 2FC            rcr     13
00648   A2B2 070            n=c
00649   A2B3 24A            c=a-c   r(-
00650   A2B4 3EE            lshfa   all
00651   A2B5 3EE            lshfa   all
00652   A2B6 3EE            lshfa   all
00653   A2B7 106            a=c     s&x
00654   A2B8 066            a()b    s&x
00655   A2B9 326            ?a(b    s&x
00656   A2BA 381            ?cgo    $02E0           NONEXISTENT
00656   A2BB 00B
00657   A2BC 198            c=m
00658   A2BD 106            a=c     s&x
00659   A2BE 0B0            c=n
00660   A2BF 1BC            rcr     11
00661   A2C0 0FA            c()b    m
00662   A2C1 0CE            c=b     all
```

```
00663   A2C2 2BC            rcr     7
00664   A2C3 0A6            a()c    s&x
00665   A2C4 070            n=c
00666
00667   A2C5 104            clrf    8
00668   A2C6 349            xqrel   rmrgguts
00668   A2C7 0BC
00668   A2C8 268
00669
00670   A2C9 198            c=m
00671   A2CA 106            a=c     s&x
00672   A2CB 0B0            c=n
00673   A2CC 270            ramslct
00674   A2CD 246            c=a-c   s&x
00675   A2CE 226            c=c+1   s&x
00676   A2CF 2BC            rcr     7
00677   A2D0 05E            c=0     ms
00678   A2D1 23E            c=c+1   ms
00679   A2D2 2F0            writd
00680   A2D3 046            c=0     s&x
00681   A2D4 270            ramslct
00682   A2D5 378            read    13
00683   A2D6 03C            rcr     3
00684   A2D7 1C6            a=a-c   s&x
00685   A2D8 36D            xqrel   binbcd
00685   A2D9 08C
00685   A2DA 102
00686   A2DB 128            writ    4
00687   A2DC 3E0            rtn
00688
00689
00690                ***************************************************************
00691                * LASTREG - Return the address of the last existent RAM reg.*
00692                *         - INPUT: NONE                                       *
00693                *         - OUTPUT: Last register adr in B,C S&X              *
00694                *         - USES: A,B,C, RAMSLCTED REG                        *
00695                ***************************************************************
00696
00697   A2DD 130  lastreg   ldi     $23F
00697   A2DE 23F
00698   A2DF 0E6            c()b    s&x         move adr to B
00699   A2E0 130  lstreg10  ldi     $40
00699   A2E1 040
00700   A2E2 066            a()b    s&x
00701   A2E3 246            c=a-c   s&x         sub $40 from adr
00702   A2E4 270            ramslct
00703   A2E5 0E6            c()b    s&x         save adr in B
00704   A2E6 038            read    0           get reg contents
00705   A2E7 10E            a=c     all         save in A
00706   A2E8 2A6            c=-c-1  s&x         complement part of C
00707   A2E9 2F0            writd               write it
00708   A2EA 038            read    0           read it back
00709   A2EB 2A6            c=-c-1  s&x         complement it again
00710   A2EC 36E            ?a#c    all         same as original ?
00711   A2ED 39F            jc      lstreg10    ..no, try lower adr
00712   A2EE 2F0            writd               restore original contents
00713   A2EF 0C6            c=b     s&x         adr to C
00714   A2F0 3E0            rtn
00715
```


PPC

```
00716                ***************************************************************
00717                * GETEP - Read the L8 and U2 EPROMs into a ROME block.        *
00718                *       - INPUT: X - The block number to load into.           *
00719                *       -        Y - The U2 section to read from.             *
00720                *       -        Z - The L8 section to read from.             *
00721                * Both the L8 and the U2 EPROMs are prompted for by the       *
00722                * program.                                                    *
00723                ***************************************************************
00724
00725  A2F1 090            con    $90
00726  A2F2 005            con    $05
00727  A2F3 014            con    $14
00728  A2F4 005            con    $05
00729  A2F5 007            con    $07             name: GETEP
00730
00731  A2F6 3C4   getep    st=0
00732  A2F7 008            setf   3               rtn early from initmce
00733  A2F8 36D            xqrel  findmc          PILTST and MC00506 findid
00733  A2F9 08C
00733  A2FA 126
00734  A2FB 36D            xqrel  bcdbinXF        X -) bin, 15 max value
00734  A2FC 08C
00734  A2FD 0E8
00735  A2FE 13C            rcr    8               6 left - make it an adr
00736  A2FF 266            c=c-1  s&x             set up ctr in s&x (FFF)
00737  A300 070            n=c                    adr/ctr in N
00738
00739                * Get the U2 section # - save in Q
00740  A301 130            ldi    4               burn sec # max is 3
00740  A302 004
00741  A303 0E6            c()b   s&x
00742  A304 0B8            read   2               get U2 burn section #
00743  A305 10E            a=c    all
00744  A306 36D            xqrel  bcdbin          burn sec to binary
00744  A307 08C
00744  A308 0ED
00745  A309 268            writ   9               save U2 # in Q
00746
00747                * Get the L8 section #
00748  A30A 130            ldi    2               burn sec # max is 1
00748  A30B 002
00749  A30C 0E6            c()b   s&x
00750  A30D 078            read   1               get L8 section #
00751  A30E 10E            a=c    all
00752  A30F 36D            xqrel  bcdbin          burn sec. to bin
00752  A310 08C
00752  A311 0ED
00753
00754
00755  A312 1BC            rcr    11              left 3, adr is 0000 or 1000
00756  A313 158            m=c                    start adr to M
00757  A314 266            c=c-1  s&x             end adr (xFFF)
00758  A315 10E            a=c    all             end adr to A
00759
00760
00761  A316 36D            xqrel  initmce
00761  A317 08C
00761  A318 037
00762
00763
00764  A319 3D9            ?ncxq  $07F6           turn on display
00764  A31A 01C
```

```
00765   A31B 3BD              ?ncxq   $07EF           mssg to display
00765   A31C 01C
00766   A31D 012              con     $12
00767   A31E 004              con     $04
00768   A31F 019              con     $19
00769   A320 020              con     $20
00770   A321 00C              con     $0C
00771   A322 038              con     $38
00772   A323 020              con     $20
00773   A324 005              con     $05
00774   A325 010              con     $10
00775   A326 012              con     $12
00776   A327 00F              con     $0F
00777   A328 20D              con     $20D            mssg: RDY L8 EPROM
00778   A329 104              clrf    8
00779   A32A 349              xqrel   gepwait
00779   A32B 08C
00779   A32C 392
00780
00781                 * Set MC00506 as talker, and source SDA
00782   A32D 2C9              ?ncxq   $70B2           TAD r5
00782   A32E 1C0
00783   A32F 3A9              ?ncxq   $70EA           SDA, wait for FRAV
00783   A330 1C0
00784
00785   A331 0B0              c=n                     get adr/ctr
00786   A332 10E              a=c     all             adr/ctr to A
00787   A333 09A              b=a     m               adr to B
00788   A334 013              jnc     rd120
00789
00790                 * Main Loop for Read1
00791   A335 280    rd110     ilw     2               retransmit the DAB
00792   A336 041    rd120     ?ncxq   $7110           get a DAB
00792   A337 1C4
00793   A338 39D              ?ncxq   $77E7           XMIT ERR on non-dab
00793   A339 1DC
00794   A33A 0EE              c()b    all             DAB to B, ROM adr to C
00795   A33B 330              fetch                   get ROM word
00796   A33C 0F6              c()b    xs              put U2 with new DAB in B
00797   A33D 0C6              c=b     s&x             new ROM word to C
00798   A33E 040              writr                   write new ROM word
00799   A33F 23A              c=c+1   m               incr ROM adr
00800   A340 0FA              c()b    m               save ROM adr in B
00801   A341 1A6              a=a-1   s&x             decr the ctr
00802   A342 39B              jnc     rd110           repeat for 4k block
00803
00804   A343 3D9              ?ncxq   $70F6           NRD and retrans last DAB
00804   A344 1C0
00805   A345 39D              ?ncxq   $77E7           XMIT ERR ??
00805   A346 1DC
00806
00807
00808
00809   A347 3C4              st=0
00810   A348 008              setf    3               rtn early from initmce
00811   A349 0DA              c=b     m               get the block adr
00812   A34A 15C              r=      6               pt to MS adr digit
00813   A34B 262              c=c-1   @r              set to correct block (n000)
00814   A34C 130              ldi     $3FF            set up ctr in s&x (3FF)
00814   A34D 3FF
00815   A34E 070              n=c                     adr/ctr in N
00816   A34F 278              read    9               get U2 sec # from Q
00817
```

```
00818                   * Make EPROM start and end adrs
00819   A350 1E6                c=c+c   s&x
00820   A351 1E6                c=c+c   s&x             0,1,2,3 -> 0,4,8,C
00821   A352 37C                rcr     12              left 2, adr is 0000 to 0C00
00822   A353 158                m=c                     start adr to M
00823   A354 10E                a=c     all             adr to A
00824   A355 130                ldi     $3FF
00824   A356 3FF
00825   A357 146                a=a+c   s&x             end = start + 3FF
00826
00827                   * Init the MC00506 - RTN after setting the range
00828   A358 36D                xqrel   initmce
00828   A359 08C
00828   A35A 037
00829
00830                   * Display prompt, beep, wait for key
00831   A35B 3D9                ?ncxq   $07F6           enable display
00831   A35C 01C
00832   A35D 3BD                ?ncxq   $07EF           chars to disp
00832   A35E 01C
00833   A35F 012                con     $12
00834   A360 004                con     $04
00835   A361 019                con     $19
00836   A362 020                con     $20
00837   A363 015                con     $15
00838   A364 032                con     $32
00839   A365 020                con     $20
00840   A366 005                con     $05
00841   A367 010                con     $10
00842   A368 012                con     $12
00843   A369 00F                con     $0F
00844   A36A 20D                con     $20D            msg: RDY U2 EPROM
00845   A36B 108                setf    8
00846   A36C 349                xqrel   gepwait
00846   A36D 08C
00846   A36E 392
00847
00848                   * Set MC00506 as talker, and source SDA
00849   A36F 2C9                ?ncxq   $70B2           TAD r5
00849   A370 1C0
00850   A371 3A9                ?ncxq   $70EA           SDA, wait for FRAV
00850   A372 1C0
00851
00852   A373 0B0                c=n                     get adr/ctr
00853   A374 10E                a=c     all             adr/ctr to A
00854   A375 09A                b=a     m               adr to B
00855   A376 013                jnc     rdu20
00856
00857                   * Main loop for Readu
00858   A377 280        rdu10   ilw     2               retransmit the DAB
00859   A378 041        rdu20   ?ncxq   $7110          get a Loop DAB
00859   A379 1C4
00860   A37A 39D                ?ncxq   $77E7           XMIT ERR on non-dab
00860   A37B 1DC
00861   A37C 37C                rcr     12              byte left 2 digits
00862   A37D 05C                r=      4               loop ctr
00863   A37E 0EE        rdu30   c()b    all             adr to C
00864   A37F 330                fetch                   get current ROM word
00865   A380 0D6                c=b     xs              get u2 bits
00866   A381 040                writr                   write new ROM word
00867   A382 23A                c=c+1   m               incr adr
00868   A383 0EE                c()b    all             pkd u2 bits to C/adr to B
00869   A384 1EE                c=c+c   all
00870   A385 1EE                c=c+c   all
```

```
00871   A386 33C            rcr     1           shift right 2 bits
00872   A387 3D4            r=r-1               decr loop ctr
00873   A388 394            ?r=     0
00874   A389 3AB            jnc     rdu30       do all 4 words
00875   A38A 1A6            a=a-1   s&x         decr outer loop ctr
00876   A38B 363            jnc     rdu10       Main loop
00877
00878   A38C 3D9            ?ncxq   $70F6       NRD and retrans last DAB
00878   A38D 1C0
00879   A38E 39D            ?ncxq   $77E7       XMIT ERR ??
00879   A38F 1DC
00880   A390 2B1            ?ncgo   $70AC       UNT and return
00880   A391 1C2
00881
00882   A392 149  gepwait   ?ncxq   $0952       deselect display
00882   A393 024
00883   A394 36D            ?ncxq   $16DB       tone 7
00883   A395 058
00884   A396 04E            c=0     all
00885   A397 05C            r=      4
00886   A398 390            ld@r    14
00887   A399 05C            r=      4
00888
00889   A39A 3C8  gepw10    clrkey
00890   A39B 22A            c=c+1   r(-         add to ctr
00891   A39C 067            jc      gepoff
00892   A39D 3CC            ?key                key down ?
00893   A39E 3E3            jnc     gepw10      wait for key
00894
00895   A39F 220            c=key               get the key
00896   A3A0 01C            r=      3
00897   A3A1 1E2            c=c+c   @r          gen carry if key is 'ON'
00898   A3A2 037            jc      gepoff
00899
00900   A3A3 3C1            ?ncxq   $2CF0       clear display
00900   A3A4 0B0
00901   A3A5 149            ?ncxq   $0952       deselect display
00901   A3A6 024
00902   A3A7 3E0            rtn                 rtn for any other key
00903
00904   A3A8 10C  gepoff    ?fset   8
00905   A3A9 01F            jc      gepclr      clear block if f8 set
00906   A3AA 3C1  gepoff10  ?ncgo   $00F0
00906   A3AB 002
00907
00908   A3AC 00E  gepclr    a=0     all
00909   A3AD 1A6            a=a-1   s&x         A = FFF (ctr)
00910   A3AE 0B0            c=n                 get adr to C
00911   A3AF 046            c=0     s&x
00912   A3B0 040  gepclr10  writr
00913   A3B1 23A            c=c+1   m           incr adr
00914   A3B2 1A6            a=a-1   s&x         decr ctr
00915   A3B3 3EB            jnc     gepclr10    clear ROM
00916   A3B4 3B3            jnc     gepoff10
00917
00918             ************************************************************
00919             *                                                          *
00920             * KASN - Key Assign (Similar to 1K in PPC ROM)             *
00921             *                                                          *
00922             *  Input:  Z - First assignment byte                       *
00923             *          Y - Second assignment byte                      *
00924             *          X - Keycode                                     *
00925             *  If both Y & Z are zero, the assignment to the key (if any)*
00926             *  is cleared.                                             *
00927             ************************************************************
```




```
00928
00929
00930   A3B5 08E              con     $8E
00931   A3B6 013              con     $13
00932   A3B7 001              con     $01
00933   A3B8 00B              con     $0B             name: KASN
00934
00935   A3B9 0F8  kasn        read    3               get keycode from X
00936   A3BA 2F6              ?c#0    xs              check for keycode < 0
00937   A3BB 01B              jnc     kasn10
00938
00939   A3BC 0B5  kasnerr     ?ncgo   $282d          "DATA ERROR"
00939   A3BD 0A2
00940
00941                         * This section of code lifted from PASN in X-Functions
00942   A3BE 10E  kasn10      a=c     all
00943   A3BF 266              c=c-1   s&x
00944   A3C0 3E7              jc      kasnerr
00945   A3C1 266              c=c-1   s&x
00946   A3C2 3D3              jnc     kasnerr
00947   A3C3 1BC              rcr     11
00948   A3C4 106              a=c     s&x
00949   A3C5 39C              r=      0
00950   A3C6 2E2              ?c#0    @r
00951   A3C7 3AB              jnc     kasnerr
00952   A3C8 130              ldi     $90
00952   A3C9 090
00953   A3CA 31C              r=      1
00954   A3CB 302              ?a<c    @r
00955   A3CC 383              jnc     kasnerr
00956   A3CD 130              ldi     $46
00956   A3CE 046
00957   A3CF 302              ?a<c    @r
00958   A3D0 017              jc      kasn20
00959   A3D1 266              c=c-1   s&x
00960   A3D2 39C  kasn20      r=      0
00961   A3D3 302              ?a<c    @r
00962   A3D4 343              jnc     kasnerr
00963   A3D5 050              ld@r    1
00964   A3D6 31C              r=      1
00965   A3D7 362              ?a#c    @r
00966   A3D8 027              jc      kasn30
00967   A3D9 36A              ?a#c    r<-
00968   A3DA 013              jnc     kasn30
00969   A3DB 16A              a=a+1   r<-
00970   A3DC 1A6  kasn30      a=a-1   s&x
00971   A3DD 0A6              a<>c    s&x
00972   A3DE 106              a=c     s&x
00973   A3DF 3C6              rshfc   s&x
00974   A3E0 3E6              lshfa   s&x
00975   A3E1 0A2              a<>c    @r
00976   A3E2 106              a=c     s&x
00977   A3E3 35E              ?a#0    ms
00978   A3E4 023              jnc     kasn40
00979   A3E5 130              ldi     8
00979   A3E6 008
00980   A3E7 146              a=a+c   s&x             binary keycode in A
00981
00982                         *put keycode in status reg 10
00983   A3E8 2B8  kasn40      read    10
00984   A3E9 0AA              a<>c    r<-
00985   A3EA 2A8              writ    10
00986
```

```
00987                   *convert the 2 assignment bytes to binary
00988   A3EB 130                ldi     256             bytes must be < 256
00988   A3EC 100
00989   A3ED 0E6                c()b    s&x             compare value in B
00990   A3EE 078                read    1               get ms FP value from Z
00991   A3EF 10E                a=c     all             move to A
00992   A3F0 36D                xqrel   bcdbin          convert to bin < 256
00992   A3F1 08C
00992   A3F2 0ED
00993   A3F3 37C                rcr     12              left 2 digits
00994   A3F4 158                m=c                     save ms byte in M
00995   A3F5 0B8                read    2               get ls FP value from Y
00996   A3F6 10E                a=c     all             move to A
00997   A3F7 36D                xqrel   bcdbin          convert to bin < 256
00997   A3F8 08C
00997   A3F9 0ED
00998   A3FA 198                c=m                     get ms value
00999   A3FB 31C                r=      1               set field
01000   A3FC 0AA                a()c    r(-             combine low & high bytes
01001   A3FD 0AE                a()c    all             save 2 bytes in A
01002
01003   A3FE 2B8                read    10              get the binary keycode
01004   A3FF 0AE                a()c    all             keycode in A / 2 bytes in C
01005   A400 01C                r=      3               field is lower 2 bytes
01006   A401 2EA                ?c#0    r(-             both assignment bytes = 0 ??
01007   A402 331                ?ncgo   $27cc           ..yes, clear the key
01007   A403 09E
01008   A404 0FC                rcr     10              move 2 bytes to digits 4-7
01009   A405 208                setf    2
01010   A406 1DD                ?ncgo   $2777           mainframe key assignment
01010   A407 09E
01011
01012
01013
01014
01015                   ************************************************************
01016                   *                                                          *
01017                   * PCAT - Port Addressable Catalog 2                        *
01018                   *                                                          *
01019                   *    Non-programmable function that prompts for a single   *
01020                   *    digit in the range 0-9. This is interpreted as a ROM  *
01021                   *    address at which to start the CAT 2 list.             *
01022                   *                                                          *
01023                   *              1  -  Port 1 (Address $8000)                *
01024                   *              2  -  Port 2 (Address $A000)                *
01025                   *              3  -  Port 3 (Address $C000)                *
01026                   *              4  -  Port 4 (Address $E000)                *
01027                   *              5  -  Timer ROM (Address $5000)             *
01028                   *              6  -  Printer ROM (Address $6000)           *
01029                   *              7  -  HPIL ROM (Address $7000)              *
01030                   *          0,8,9  -  Normal CAT 2 (Address $5000)          *
01031                   *                                                          *
01032                   * Note:  This function prints strangely on the printer when *
01033                   *        in NORM or TRACE modes. This is a bug in the HP41 *
01034                   *        mainframe. It can only be avoided by doing the digit *
01035                   *        prompt from within this routine, rather the using *
01036                   *        the built-in prompting.                           *
01037                   ************************************************************
01038
01039   A408 094                con     $094
01040   A409 001                con     $001
01041   A40A 303                con     $303
01042   A40B 110                con     $110            name: PCAT (With 1 dig. prompt
)
```


PPC

```
01043
01044   A40C 000   pcat      nop                 non-programmable
01045   A40D 0AE             a()c    all         port # to C
01046   A40E 358             st=c                move to ST
01047   A40F 151             ?ncxq   $0054       check & process indirect
01047   A410 000
01048   A411 398             c=st                get port #
01049   A412 39C             r=      0           set field
01050   A413 1E2             c=c+c   @r          double port #
01051   A414 0BF             jc      pcdflt      default on 8 or 9
01052   A415 2E6             ?c#0    s&x         ck for port 0
01053   A416 0AB             jnc     pcdflt      default on 0
01054   A417 106             a=c     s&x         doubled port # to A
01055   A418 130             ldi     6
01055   A419 006
01056   A41A 202             c=c+a   @r          +6 to get port adr for 1-4
01057   A41B 017             jc      pcat10      if port was 5,6,7 use original
01058   A41C 358             st=c                save port adr in ST
01059   A41D 04E   pcat10    c=0     all         clear C
01060   A41E 15C             r=      5           set ptr
01061   A41F 398             c=st                get port adr
01062   A420 13C             rcr     8           port adr to digit 6
01063   A421 10E             a=c     all         port adr in d6 of A/ S&X clear
01064   A422 150             ld@r    5           start adr = 5000
01065   A423 23A             c=c+1   m           adr = 5001
01066   A424 15C             r=      6           reset ptr
01067
01068   A425 362   pcat20    ?a#c    @r          test port adr
01069   A426 033             jnc     pcat30      jump if equal
01070   A427 330             fetch               get # of items in cat
01071   A428 146             a=a+c   s&x         accumulate in A
01072   A429 222             c=c+1   @r          incr to next ROM
01073   A42A 3DB             jnc     pcat20      loop
01074
01075   A42B 00E   pcdflt    a=0     all
01076   A42C 130   pcat30    ldi     2           cat '2'
01076   A42D 002
01077   A42E 1BC             rcr     11          move to digit 3
01078   A42F 11A             a=c     m           move to A digit 3
01079   A430 238             read    8           get the P reg
01080   A431 0AE             a()c    all
01081   A432 07C             rcr     4           move 2abc to 4 MS digits of C
01082   A433 09C             r=      5
01083   A434 0AA             a()c    r(-         put ALPHA from P into C
01084   A435 239             ?ncgo   $0b8e       jump into catalog function
01084   A436 02E
01085
01086
01087
01088
01089
01090
01091   ************************************************************
01092   * Init MC00505A - Assume: Loop addressed, MC00506 in r5    *
01093   *                 - REN to get ready for CMDS              *
01094   *                 - Listen the primary device              *
01095   *                 - SDC to reset the device                *
01096   *                 - Send R, D, Y, B (each followed by LF)  *
01097   *                 - Send the start adr (in M d3-0)         *
01098   *                 - Send the term. adr (in A d3-0)         *
01099   *                 - RTN if F3 set                          *
01100   *                 - Do a verify blank (ERASE ERR if not)   *
01101   *                 - Send start adr (in M d3-0)             *
01102   *                 - Send NRE (get ready for data)          *
01103   ************************************************************
```

```
01104   A437          LF        equ                   10
01105
01106   A437 130      initmce   ldi     $092
01106   A438 092
01107   A439 2E9                ?ncxq   $70BA          REN command
01107   A43A 1C0
01108   A43B 375                ?ncxq   $70DD          LAD to r5
01108   A43C 1C0
01109   A43D 130                ldi     $004
01109   A43E 004
01110   A43F 2E9                ?ncxq   $70BA          SDC command
01110   A440 1C0
01111   A441 36D                xqrel   ildata         send string to loop
01111   A442 08C
01111   A443 096
01112   A444 052                con     $52            'R' reset
01113   A445 00A                con     LF
01114   A446 044                con     $44            'D' disable CRLF
01115   A447 00A                con     LF
01116   A448 059                con     $59            'Y' 1 status byte
01117   A449 00A                con     LF
01118   A44A 042                con     $42            'B' binary mode
01119   A44B 00A                con     LF
01120
01121
01122                   * Send start address (SbbLF)
01123   A44C 253                con     $253           'S' - end of string
01124   A44D 198                c=m                    get address from M
01125   A44E 23C                rcr     2              MS byte in C d1,0
01126   A44F 0A1                ?ncxq   $7128          send loop DAB
01126   A450 1C4
01127   A451 198                c=m                    LS byte in C d1,0
01128   A452 0A1                ?ncxq   $7128          send 2nd adr byte
01128   A453 1C4
01129   A454 36D                xqrel   ildata         send string to loop
01129   A455 08C
01129   A456 096
01130   A457 00A                con     LF             punctuate start adr
01131
01132                   * Send term address (TbbLF)
01133   A458 254                con     $254           'T' - end of string
01134   A459 0AE                a()c    all
01135   A45A 10E                a=c     all            copy end adr from A d3-0
01136   A45B 23C                rcr     2              MS byte in C d1,0
01137   A45C 0A1                ?ncxq   $7128          DAB
01137   A45D 1C4
01138   A45E 0AE                a()c    all            get end adr from A
01139   A45F 0A1                ?ncxq   $7128          DAB
01139   A460 1C4
01140   A461 130                ldi     LF
01140   A462 00A
01141   A463 0A1                ?ncxq   $7128          DAB
01141   A464 1C4
01142   A465 39D                ?ncxq   $77E7          error check
01142   A466 1DC
01143   A467 00C                ?fset   3
01144   A468 360                ?crtn                  RTN if F3 set
01145
01146                   * Verify that this address range is blank
01147   A469 36D                xqrel   ildata         send string to loop
01147   A46A 08C
01147   A46B 096
01148   A46C 056                con     $56            'V' - verify blank
01149   A46D 20A                con     $20A           LF - end of string
```




```
01150   A46E 2C9            ?ncxq   $70B2       TAD to primary device
01150   A46F 1C0
01151   A470 130            ldi     $061        SST byte
01151   A471 061
01152   A472 0D9            ?ncxq   $7136       Send SST, get DAB in A
01152   A473 1C4
01153   A474 39D            ?ncxq   $77E7       do err ck
01153   A475 1DC
01154   A476 386            rshfa   s&x         move stat byte to A d1,0
01155   A477 0A6            a()c    s&x
01156   A478 3D8            c()st               status byte to ST
01157   A479 00C            ?fset   3           test for bad verify
01158   A47A 023            jnc     imce10      ..no, continue
01159   A47B 341            gorel   eraserr     ERASE ERR
01159   A47C 08C
01159   A47D 13A
01160   A47E 3D8  imce10    c()st               replace ST
01161
01162
01163                       * Send start adr again
01164   A47F 375            ?ncxq   $70DD       LAD to primary device
01164   A480 1C0
01165   A481 130            ldi     $53         'S' char
01165   A482 053
01166   A483 099            ?ncxq   $7126       DAB
01166   A484 1C4
01167   A485 198            c=m                 get address from M
01168   A486 23C            rcr     2           MS byte in C d1,0
01169   A487 0A1            ?ncxq   $7128       send loop DAB
01169   A488 1C4
01170   A489 198            c=m                 LS byte in C d1,0
01171   A48A 0A1            ?ncxq   $7128       send 2nd adr byte
01171   A48B 1C4
01172   A48C 130            ldi     LF
01172   A48D 00A
01173   A48E 0A1            ?ncxq   $7128       DAB
01173   A48F 1C4
01174
01175                       * Send NRE so subsequent data bytes may be sent
01176   A490 130            ldi     $093
01176   A491 093
01177   A492 2E9            ?ncxq   $70BA       send NRE cmd
01177   A493 1C0
01178   A494 39D            ?ncgo   $77E7       do the error check and return

01178   A495 1DE
01179
01180
01181
01182                       ********************************************************
01183                       * ildata - Send the string of bytes after the XQ to the loop *
01184                       *        - Terminates on byte whose XS is not 0        *
01185                       *        - Returns following the byte string           *
01186                       *        - Loop is assumed to be addressed, with 41 as talker*
01187                       ********************************************************
01188   A496 1B0  ildata    pop                 get the rtn adr
01189   A497 330            fetch               fetch the data byte there
01190   A498 23A            c=c+1   m           incr adr
01191   A499 170            push                back on rtn stk
01192   A49A 2F6            ?c#0    xs          is this last DAB
01193   A49B 099            ?cgo    $7126       ..yes, rtn after DAB
01193   A49C 1C7
01194   A49D 099            ?ncxq   $7126       ..no, continue doing DABs
01194   A49E 1C4
01195   A49F 3BB            jnc     ildata
01196
01197
```

```
01198                ***************************************************************
01199                * DISPNAME -  Display the name of the function whose entry    *
01200                *             point is in the adr field of A.                 *
01201                *          -  The name MUST be 6 or less chars.               *
01202                *          -  The last char of the name gets a ':' added.     *
01203                *          -  The name is right-justified in chars 1-6.       *
01204                ***************************************************************
01205
01206   A4A0 3C1   dispname  ?mcxq  $2CF0          clear the display
01206   A4A1 0B0
01207   A4A2 0BA             a()c   m              get the function adr to C Mant
01208   A4A3 34D             ?mcxq  $05D3          function name to display
01208   A4A4 014
```

```
01256   A4C9 2D4            ?r=     13
01257   A4CA 3CB            jnc     disp20
01258   A4CB 068            writ    1               write upper 4 bits
01259
01260                   * Read lower 4 digits and reverse them
01261   A4CC 038            read    0
01262   A4CD 37C            rcr     12              left 2
01263   A4CE 35C            r=      12              loop counter
01264   A4CF 11E    disp50  a=c     ms              transfer the char
01265   A4D0 38E            rshfa   all
01266   A4D1 2FC            rcr     13              C left 1
01267   A4D2 3D4            r=r-1
01268   A4D3 394            ?r=     0
01269   A4D4 3DB            jnc     disp50
01270
01271                   * add hex adr to right 4 chars of display (lower 4 bits)
01272   A4D5 198            c=m                     get adr
01273   A4D6 07C            rcr     4               adr to d13-10
01274   A4D7 0AE            a()c    all             adr to A, dsp str to C
01275   A4D8 0BC            rcr     5               position dsp str
01276   A4D9 2DC            r=      13
01277
01278   A4DA 322    disp60  ?a(b    @r              digit ( $A?
01279   A4DB 01F            jc      disp70          ..yes, branch
01280   A4DC 182            a=a-b   @r              ..no, subtract 9
01281   A4DD 162            a=a+1   @r
01282   A4DE 0A2    disp70  a()c    @r              adr char to disp
01283   A4DF 3AE            rshfb   all             move the $A down
01284   A4E0 3D4            r=r-1
01285   A4E1 254            ?r=     9               process digits 13-10
01286   A4E2 3C3            jnc     disp60
01287   A4E3 0FC            rcr     10
01288   A4E4 028            writ    0               write lower 4 bits to dis
01289   A4E5 149            ?ncxq   $0952           deselect the display
01289   A4E6 024
01290   A4E7 3E0            rtn
01291
01292                   ************************************************************
01293                   *BCDBIN - Decimal to Binary conversion                       *
01294                   *      - INPUT: A - FP number to convert                    *
01295                   *               B - Maximum number to convert in S&X        *
01296                   *      - OUTPUT: Binary in A & C S&X                        *
01297                   *               A&C are zero except the binary in S&X       *
01298                   *      - USES:  A,C,B S&X,F8                                *
01299                   * Tests for ALPHA DATA, and DATA ERROR on values greater than *
01300                   * 999 decimal or the number in B.                           *
01301                   * BCDBINX does the conversion on the X reg (rather than A).  *
01302                   * BCDBINXF does the conversion on X with limit of 15 (0kF).  *
01303                   ************************************************************
01304
01305   A4E8 130    bcdbinXF ldi    16              max value is 15
01305   A4E9 010
01305   A4EA 0E6            c()b    s&x
01307
01308   A4EB 0F8    bcdbinX  read   3               get the X register
01309   A4EC 10E             a=c    all
01310
01311   A4ED 0AE    bcdbin   a()c   all             FP to C
01312   A4EE 361             ?ncxq  $14D8           ALPHA DATA check
01312   A4EF 050
01313   A4F0 260             sethex
01314   A4F1 10E             a=c    all
01315   A4F2 356             ?a#0   xs              is XP negative
```

```
01316   A4F3 03F              jc      gobcdbin        ..yes
01317   A4F4 130              ldi     3
01317   A4F5 003
01318   A4F6 306              ?a(c    s&x             XP less than 3 ?
01319   A4F7 0B5    bcdDE     ?ncgo   $282D           ..no, DATA ERROR
01319   A4F8 0A2
01320   A4F9 0AE              a()c    all             original FP to C
01321   A4FA 39D    gobcdbin  ?ncxq   $02E7          mainframe BCDBIN
01321   A4FB 008
01322   A4FC 05E              c=0     ms
01323   A4FD 05A              c=0     m               clr C except s&x
01324   A4FE 106              a=c     s&x             return # in A & C
01325   A4FF 326              ?a(b    s&x             # ( B ?
01326   A500 3BB              jnc     bcdDE           ..no, DATA ERROR
01327   A501 3E0              rtn
01328
01329                 ****************************************************************
01330                 * BINBCD - Binary to floating point decimal conversion.        *
01331                 *        - INPUT: Binary number in A S&X (up to FFF hex).      *
01332                 *        - OUTPUT: Floating point equiv in C.                  *
01333                 *        - USES: A,C,R - returns in Hex mode.                  *
01334                 ****************************************************************
01335
01336   A502 2A0    binbcd    setdec
01337   A503 0A6              a()c    s&x
01338   A504 106              a=c     s&x
01339   A505 01A              a=0     m
01340   A506 25C              r=      9
01341   A507 033              jnc     binbcd1
01342   A508 1FA    binbcd0   c=c+c   m
01343   A509 1FA              c=c+c   m
01344   A50A 1FA              c=c+c   m
01345   A50B 1FA              c=c+c   m
01346   A50C 11A              a=c     m
01347   A50D 05A    binbcd1   c=0     m
01348   A50E 2FC              rcr     13
01349   A50F 23A              c=c+1   m
01350   A510 27A              c=c-1   m
01351   A511 21A              c=c+a   m
01352   A512 3DC              r=r+1
01353   A513 354              ?r=     12
01354   A514 3A3              jnc     binbcd0
01355   A515 260              sethex
01356   A516 01B              jnc     binbcd2
01357   A517 35C              r=      12
01358   A518 0AE              a()c    all
01359   A519 130    binbcd2   ldi     $009
01359   A51A 009
01360   A51B 11A              a=c     m
01361   A51C 2FA              ?c#0    m
01362   A51D 02F              jc      binbcd4
01363   A51E 04E              c=0     all
01364   A51F 3E0              rtn
01365   A520 266    binbcd3   c=c-1   s&x
01366   A521 3FA              lshfa   m
01367   A522 342    binbcd4   ?a#0    @r
01368   A523 3EB              jnc     binbcd3
01369   A524 0BA              a()c    m
01370   A525 3E0              rtn
01371
```

```
01372                ********************************************************************
01373                * FINDID for MC00506A                                              *
01374                * Initializes the loop.                                            *
01375                * Searches from the primary device, leaving the first              *
01376                *   MC00506A found as the selected device in PIL register 5.       *
01377                *   This device is also left addressed as a talker.                *
01378                *                                                                  *
01379                * USES: A,B S&X ,C,F8,F9,R, 3 Subroutine levels.                   *
01380                ********************************************************************
01381
01382  A526 36D   findmc    xqrel   piltst          Test for existence of PIL
01382  A527 08C
01382  A528 176
01383  A529 18D             ?ncxq   $7063           address the loop
01383  A52A 1C0
01384  A52B 39D             ?ncxq   $77E7           check XMIT err
01384  A52C 1DC
01385  A52D 086             b=a     s&x             save # of devices in B
01386  A52E 155             ?ncxq   $7155           set primary device in r5, r6
01386  A52F 1C4
01387  A530 104             clrf    8
01388  A531 2C9   fndmc10   ?ncxq   $70B2           TAD (r5)
01388  A532 1C0
01389  A533 064             selp    4
01390  A534 285             con     $285            RDY type frame
01391  A535 0A4             selp    5
01392  A536 189             con     $189            SDI frame
01393  A537 31C             r=      1               field defn
01394  A538 36D             xqrel   fndmc30         push adr of string
01394  A539 08C
01394  A53A 144
01395  A53B 04D             con     $4D
01396  A53C 043             con     $43
01397  A53D 030             con     $30
01398  A53E 030             con     $30
01399  A53F 035             con     $35
01400  A540 030             con     $30
01401  A541 236             con     $236            last char: MC00506
01402
01403  A542 0A6   fndmc20   a<>c     s&x             get back the DAB
01404  A543 280             ilw     2               retransmit it
01405  A544 244   fndmc30   clrf    9
01406  A545 041             ?ncxq   $7110           get a DAB
01406  A546 1C4
01407  A547 24C             ?fset   9               set if frame not DAB
01408  A548 0B7             jc      fmcnext         try next device
01409  A549 106             a=c     s&x             save DAB in A
01410  A54A 1B0             pop                     get the current table adr
01411  A54B 330             fetch                   get the table character
01412  A54C 23A             c=c+1   m               incr table adr
01413  A54D 170             push                    save it for next time
01414  A54E 36A             ?a#c    r<-             DAB same as table ?
01415  A54F 027             jc      fmcnrd          ..no, send NRD
01416  A550 2F6             ?c#0    xs              end of table ?
01417  A551 38B             jnc     fndmc20         ..no, match next char
01418  A552 108             setf    8               indicate device found
01419  A553 0A6   fmcnrd    a<>c    s&x             get DAB back to C
01420  A554 020             xq->go                  pop the rtn adr
01421  A555 3D9             ?ncxq   $70F6           NRD
01421  A556 1C0
01422  A557 013             jnc     *+02            skip the extra pop off
01423  A558 020   fmcnext   xq->go                  pop the rtn adr
01424  A559 10C             ?fset   8
```

```
01425   A55A 360            ?crtn                RTN if MC00506A was found
01426   A55B 066            a()b    s&x
01427   A55C 086            b=a     s&x          copy # of devices to A
01428   A55D 185            ?ncxq   $7161        try next device (if AUTOID)
01428   A55E 1C4
01429   A55F 013            jnc     notfound     ..error - MC00506A not found
01430   A560 28B            jnc     fndmc10      SDI to next device
01431
01432   A561 2CD  notfound  ?ncxq   $77B3        mssg to disp
01432   A562 1DC
01433   A563 00E            con     $0E
01434   A564 00F            con     $0F
01435   A565 020            con     $20
01436   A566 00D            con     $0D
01437   A567 003            con     $03
01438   A568 030            con     $30
01439   A569 030            con     $30
01440   A56A 035            con     $35
01441   A56B 030            con     $30
01442   A56C 036            con     $36
01443   A56D 201            con     $201         mssg: NO MC00506A
01444   A56E 3F9            ?ncgo   $7CFE        error halt
01444   A56F 1F2
01445
01446
01447
01448             ************************************************************
01449             *ERRMSSG - Display following characters as an error mssg.   *
01450             *        - Execution returns following error string.       *
01451             *        - Functions using HPIL should use $77B3           *
01452             ************************************************************
01453
01454   A570 3A1  errmssg   ?ncxq   $22E8
01454   A571 088
01455   A572 3C1            ?ncxq   $2CF0        Clear the display
01455   A573 0B0
01456   A574 3BD            ?ncgo   $07EF        Disp the mssg and RTN
01456   A575 01E
01457
01458
01459             ************************************************************
01460             *PILTST - Test for existence of HPIL Chip.                 *
01461             *       - RTN if it exists, else "NO HPIL" error.          *
01462             *       - The test is simple - look for 28 (XROM #) at $7000*
01463             *       - USES: C, A S&X, R                                *
01464             ************************************************************
01465
01466   A576 04E  piltst    c=0     all
01467   A577 15C            r=      6
01468   A578 1D0            ld@r    7            C adr field = $7000
01469   A579 330            fetch                get the word @ $7000
01470   A57A 106            a=c     s&x          word to A
01471   A57B 130            ldi     28           XROM # is 28
01471   A57C 01C
01472   A57D 366            ?a#c    s&x          is PIL Chip existent ?
01473   A57E 3A0            ?ncrtn               ..yes, return
01474   A57F 36D            xqrel   errmssg      ..no, disp "NO HPIL"
01474   A580 08C
01474   A581 170
01475   A582 00E            con     $0E
01476   A583 00F            con     $0F
01477   A584 020            con     $20
01478   A585 008            con     $08
01479   A586 010            con     $10
01480   A587 009            con     $09
01481   A588 20C            con     $20C         mssg: NO HPIL
```




```
01482
01483
01484
01485              ***************************************************************
01486              *ERRHALT - Assumes a message in the display (ERRMSSG).        *
01487              *        - Left justifies the mssg and halts (F25 checked).   *
01488              *        - Functions using HPIL should use $7CFE (2 bytes).   *
01489              ***************************************************************
01490
01491   A589 3DD   errhalt   ?ncxq   $2BF7          Left justify display
01491   A58A 0AC
01492   A58B 108             setf    8
01493   A58C 201             ?ncxq   $1C80          Mssg to printer
01493   A58D 070
01494   A58E 3ED             ?ncgo   $22FB          Mainframe error handler
01494   A58F 08A
01495
01496
01497
01498
01499
01500
01501              **** Beginning of Free Space
01502   A590 000   begfree   con     0
01503
01504
01505
01506              **** End of Free Space
01507   AFEE                 org     $AFEE
01508   AFEE 000   endfree   con     0
01509
01510
01511   AFF0                 org     $AFF0
01512
01513              * Beg and end of free space ptrs
01514   AFF0 005             fat     begfree
01514   AFF1 090
01515   AFF2 00F             fat     endfree
01515   AFF3 0EE
01516
01517
01518
01519              * Polling points
01520
01521   AFF4 000             con     0              FF4 pause poll
01522   AFF5 000             con     0              FF5 prgm line/IO activity
01523   AFF6 000             con     0              FF6 wakeup by peripheral
01524   AFF7 000             con     0              FF7 power down (OFF key)
01525   AFF8 000             con     0              FF8 IO service flag set
01526   AFF9 000             con     0              FF9 wakeup by ON key
01527   AFFA 000             con     0              FFA memory lost
01528
01529
01530              * ROM Revision code (FFB - FFE)
01531
01532   AFFB 003             con     $03            Rev: MC-1C
01533   AFFC 031             con     $31
01534   AFFD 003             con     $03
01535   AFFE 00D             con     $0D
01536
01537
01538              * ROM checksum
01539   AFFF 217             cksm
01540

     00000 errors
     00000 warnings
     00107 symbols defined
```

